

# D<sup>3</sup>G: Novel Approaches to Data Statistics, Understanding and Preprocessing on the Grid

Alexander Wöhrer<sup>1</sup>, Lenka Nováková<sup>2</sup>, Peter Brezany<sup>1</sup>, and A Min Tjoa<sup>3</sup>

<sup>1</sup> Institute of Scientific Computing, University of Vienna  
Nordbergstrasse 15, 1090 Vienna, Austria, {woehrer|brezany}@par.univie.ac.at

<sup>2</sup> Department of Cybernetics, Czech Technical University

Technická 2, 166 27 Prague 6, Czech Republic, novakova@labe.felk.cvut.cz

<sup>3</sup> Institute of Software Technology and Interactive Systems, Vienna University of Technology  
Favoritenstrasse 9-11, 1040 Vienna, Austria, tjoa@ifs.tuwien.ac.at

## Abstract

*Relocating the code for data preprocessing (DPP) closer towards the data source is the overall task of the D<sup>3</sup>G framework (Data Statistics, Data Understanding, Data Preprocessing on the Grid), developed within a joint project of the University of Vienna, the Vienna University of Technology and the Czech Technical University. This work presents the data service side architecture to gather data statistics on-the-fly and use them in remote DPP methods on query results as well as an approach to gather exact continuous data statistics for whole tables in a database on the Grid. The performance results of our prototype implementation are showing low running costs for the continuous data statistics inside the database and also the feasibility of our proposed data service side functionality.*

## 1 Introduction and Motivation

Within the knowledge discovery process, data preprocessing (DPP) is a time consuming and important part. Based on [11], up to 60 percent of the knowledge discovery process time is spent on data preparation. There are numerous different tools and methods used for preprocessing [8], including: *sampling*, which selects a representative subset from a large population of data; *transformation*, which manipulates raw data to produce a single input; *denoising*, which removes noise from data; *normalization*, which organizes data

for more efficient access; and *feature extraction*, which pulls out specified data that is significant in some particular context.

The research effort reported in this paper introduces a Grid enabled data preprocessing strategy. It is based on the concepts of the GGF DAIS-WG [5] and the exploitation of RDBMS (relational database management system) capabilities to provide exact continuous data statistics. The data statistics represent additional metadata of a data source, providing new views on it. The urgent need arises from two facts: first, some data of our pilot application about traumatic brain injury patients [3] for our GridMiner [15] project is missing and/or is noisy and second, the absence of non-proprietary Grid-aware data preprocessing tools for relational data. This work contributes to the overall target of the GridMiner project to cover all aspects of the knowledge discovery process and integrates them into an advanced service-oriented Grid application. With no quality data, there can be no quality data mining results (whereas the impact of dirty data on the mining results depends on the used data mining technique). The requirements on such DPP services include: *standard-based*, to easily work together with third-party tools e.g. for visualization; *extendable*, to include special and just locally needed functionality, and *flexible*, to support various usage scenarios. The list for the functionality provided via the RDBMS includes: *configurable*, in order to tailor to various needs and *low running costs* for little cumbering the RDBMS. To this end, we present an architecture for the analysis (via data statistics) and subsequent

remote application of data preprocessing methods (able to use the already gathered data statistics) on query results inside a Grid Data Service (GDS). Additionally, it provides access to incrementally maintained data statistics of pre-defined database tables via the representing GDS. Taking into account the work of [7, 16] pointing out the high costs of moving data, our approach of relocating DPP functionality towards the GDS and inside the RDBMS offers the following advantages: reducing the transportation costs by *allowing to work remotely* with the results of a query request on the data providing GDS (likely to be as close to the data source as possible); incrementally maintained data statistics tailored to data mining help to choose among potential data sources; caching query results at the GDS (together with their statistics) for later usage reduces the number of data source accesses.

The rest of the paper is organized as follows. Section 2 introduces usage scenarios for a GDS enhanced with DPP functionalities. The proposed architecture of D<sup>3</sup>G is described in Section 3, whereas Section 4 characterizes the implementation of it followed by performance results in Section 5. We are discussing related work in Section 6 and finishing our work with conclusions and a future work outline in Section 7.

## 2 Novel Usage Scenarios

The scenarios in Figure 1 and Figure 2 are showing different usage modes of *relocated data preprocessing tasks* towards the Grid Data Service. For the first use case illustrated in Figure 1, imagine the application of three (possibly) different DPP techniques to the same result set of a

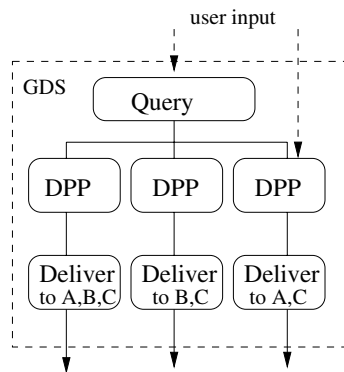


Figure 1. Active usage mode of DPP

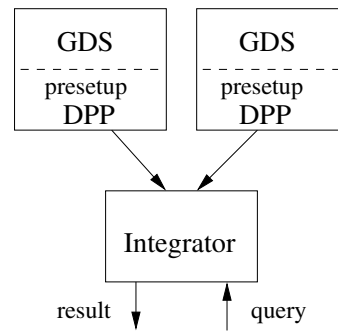


Figure 2. Passive usage mode of DPP

SQL query. We call this *active DPP*, because the requesting client specifies the query and sets the parameters for the DPP method itself. He keeps the control of what data he wants and how to preprocess it. After the preprocessing, the results are delivered over different mechanisms (various once are supported by OGSA-DAI, e.g. FTP and streaming) to the data miners called A, B and C. In general, the advantage of this approach is that the *code is very close to the data*, resulting in less bandwidth usage and latency compared to a DPP service residing on a remote computing node. It saves integration work/iteration cycles for tuning the DPP by reusing the data for multiple parallel DPP tasks.

The second use case presented in Figure 2 points out the importance of local knowledge which is required for effective DPP. In this way, the GDS can offer *various cleaned data views* for different data requestors. We call this *passive DPP*, because the requesting client (or Integrator) gets preprocessed results from earlier (locally) setup DPP methods. This can be based on the clients first-hand usage experience or local expert knowledge about the data source.

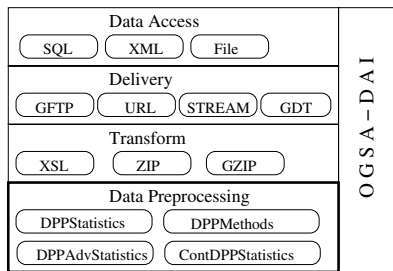
However, means to access the unaltered data - if allowed by the local security policies - have to be provided as well as information about the applied preprocessing methods with parameters. Preferable, the user should have the possibility to inspect the processes to be applied in advance. An additional important point is the documentation of how a data set has been generated, often called data provenance or lineage. This has to be explicitly bound to the result data set, because if the setup of the predefined DPP methods changes, this meta-data is lost.

### 3 The Architecture

OGSA-DAI [10] is the de-facto standard middleware to assist with access and integration of data from separate data sources within the Grid. It is based on an engine that can execute chain of activities (representing the basic building blocks of OGSA-DAI and are designed to inter-operate) specified in a perform document (XML interface). With our research effort addressing the development and integration of data preprocessing functionality into OGSA-DAI and the underlying RDBMS, we are addressing a common data mining issue – unfortunately widely ignored – in the data access research community, not only in Grid research. Especially where data integration is needed (which is often the cause of dirty data) and good data is a *prerequisite* for producing effective models – a flexible, reusable and open data preprocessing tool is required. For this purpose we:

1. *extended OGSA-DAI* with our own customized activities to compute basic data statistics about query results. Additional information about the result set can be provided to get advanced data statistics. These statistics can be used to perform remote DPP on query results.
2. *exploite RDBMS capabilities* to initially compute data statistics about a predefined set of tables by using Stored Procedures (a set of SQL operations to be compiled and executed on a database server), and maintain them incrementally, by using Triggers (a set of SQL statements that automatically 'fires off' an action when a specific event occurs).

The integrated activities focus on data preprocessing in addition to the predefined activities for



**Figure 3. OGSA-DAI predefined activities with integrated ones for DPP**

```
<UnivariateStats field="A">
  <Counts missingFreq="0" totalFreq="100">
    <Extension name="distinct" value="10"/>
  </Counts>
  <NumericInfo maximum="19.9"
    mean="15.94"
    minimum="12.03"
    standardDeviation="5.2714"/>
</UnivariateStats>
```

**Figure 4. Basic statistics**

data access (for SQL, XML and file resources), transformation (XSL and data compression) and delivery, as illustrated in Figure 3. The first activity, called *DPPStatistics*, can be seen as a metadata extraction activity. It computes basic information about a dataset in WebRowset format [14], e.g. for a numerical column A of some table, as depicted in Figure 4.

No additional input is required to gather this basic information about a dataset. All data statistics are presented to the client in PMML [4] format for interoperability. The gathered information about a dataset can be used to decide what a DPP technique to use or whether DPP is necessary at all, which is especially interesting for very expensive DPP methods. The second activity, called *DPPAdvStatistics*, allows to provide additional input information (in PMML format) about a dataset, e.g. intervals and type of attributes shown in Figure 5, allowing the advanced data statistics to include interval and histogram like information (specified in PMML).

The third activity, called *ContDPPStatistics*, provides an interface to the incrementally maintained statistics about whole tables. It returns the current data statistics about specified tables in PMML format. The fourth one, called *DPPMethods*, does the actual data preprocessing work. It

```
<DataField dataType="double" name="A"
  optype="continuous">
  <Interval closure="openOpen"
    leftMargin="12.0" rightMargin="14.0"/>
  <Interval closure="closedOpen"
    leftMargin="14.0" rightMargin="20.0"/>
</DataField>
...
<DataField dataType="double" name="H"
  optype="categorical"/>
```

**Figure 5. Additional input to get advanced statistics**

uses the precomputed data statistics to perform its task. Figure 6 shows an example invocation, using a basic preprocessing method on some data set.

```
<dppMethods name="M1">
  ...
  <column colToApply="1">
    <methode forCause="isMissing"
      impl="ReplaceMissingValuesByValue">
      <Param>15</Param>
    </methode>
  </column>
  ...
</dppMethods>
```

Figure 6. Setup of DPP methods

## 4 Implementation

The current prototype, supporting the active usage mode, is implemented by using OGSA-DAI R5 and using the RDBMS capabilities of Oracle 10g. Both, the service implementation (for query results) as well as the RDBMS components (for whole tables), use the following functions to incrementally calculate the mean  $\bar{x}$  and standard deviation  $\sigma^2$  by one iteration over the data set:

$$\bar{x} = \frac{\bar{x}_o * n + x}{n+1}$$

$$\sigma^2 = \frac{1}{n+1} * (\sigma_o^2 * n + (x - \bar{x})^2 + n * (\bar{x} - \bar{x}_o)^2)$$

where  $n$  is the number of rows the current statistics are based on and the prefix  $_o$  indicates their current value.

The *DPPStatistics* activity requires as input a data set in WebRowset format, which is cached on the local file system, and returns the basic data statistics plus a WebRowset ID (representing the cached data set) to which they belong. A *DPPAdvStatistics* activity requires as input a valid WebRowset ID plus information about types and intervals, e.g. in Figure 5, to calculate the advanced data statistics from the cached data set for columns where additional information has been provided. The *DPPMethods* activity can work directly on an input data set (when just using DPP functions which do not require data statistics) or on a cached data set by providing the WebRowset ID. It dynamically loads the specified Java functions, e.g. a simple replace function in Figure 6, and applies all of them in one iteration over the data set. Our current implementation distinguishes between two application-cases: data is missing (basic statistics

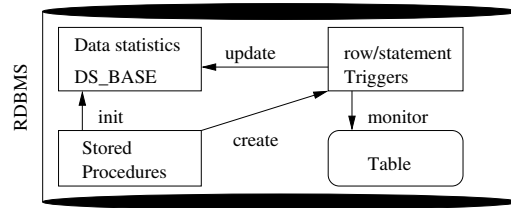


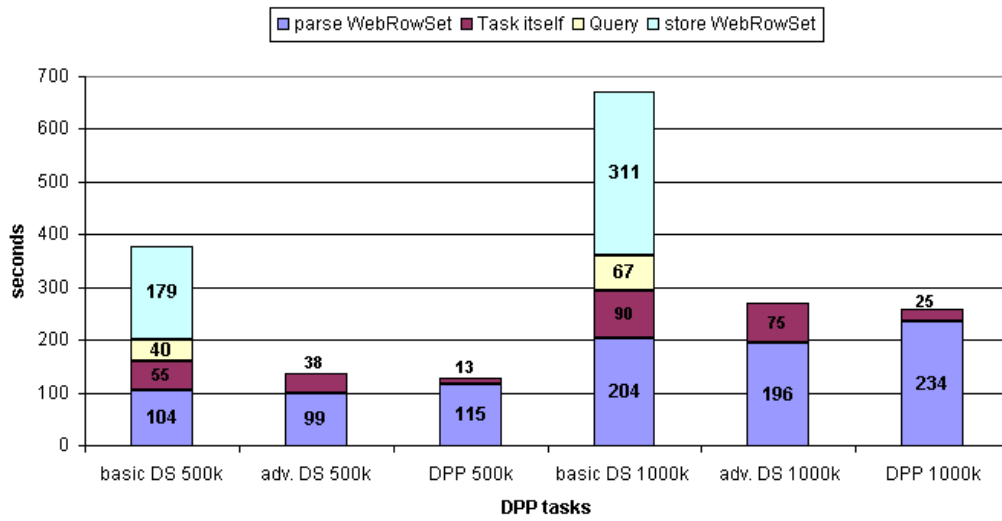
Figure 7. Components to provide exact continuous data statistics

required) and data invalid (extended statistics required). The processed data set is returned in WebRowset format. The included standard methods for DPP can easily be expanded by implementing the needed functionality by extending an abstract base class, which provides access to the data statistics of a data set as well as to the given parameters for a DPP method.

To access the continuous (basic) data statistics about a data source, no inputs are required. Figure 7 shows the implemented components and how they work together. The *ContDPPStatistics* activity queries a table where the basic data statistics are stored and transforms them into PMML. The setup for continuous data statistics works as follows: in the *DS\_CONF (tablename, schema)* table the user specifies for which table he wants data statistics. The stored procedure *init\_stats* analyzes each column of the specified tables and stores the gathered data statistics in a table called *DS\_BASIC (tablename, schema, column, min, max, mean, std-dev, missing, total\_frequency)*. To keep the statistics accurate, update- delete- and insert row triggers (fired for each row effected by a statement) are dynamically generated according to the table structure of the specified tables as well as statement triggers (fired once per statement) for update- and delete statements (to maintain the *min* and *max* values).

## 5 Preliminary Performance Results

The setup for the performance tests is as follows. OGSA-DAI R5 is deployed in a Tomcat 5 on a non-dedicated Sun Fire V880 with 8 GB RAM and 4 UltraSPARC-III 750MHz, accessing a local MySQL 4 database with about 1.000.000 rows and 11 (9 of it numerical) columns (generated by PRED0 [1]). The workstation hosting the Oracle 10g database installation, with the same data set as the other one, was a dedicated AMD 1 GHz



**Figure 8. Time spent for the various parts of an activity**

with 768 MB RAM. Figure 9 pictures the performance of our developed activities inside a standard OGSA-DAI for DPP. The standard validating Java WebRowset implementation did not scale well, so we use a non validating one for all our activities.

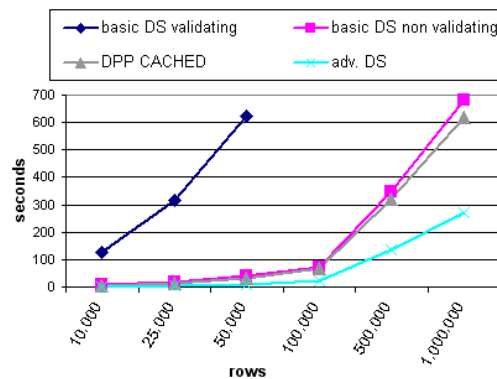
The additional information for the advanced data statistics were given for two columns, shown in Figure 5. The DPP task applied two basic methods to two different columns when the value of it was missing: once *replace missing value by mean* and once *replace missing value by given value*. A more detailed view on where the time is spent inside an activity is given in Figure 8.

	950k rows	1.900k rows
init_stats SP	4.344	10.841
insert RT	2	2
update RT	5	5
delete RT	3	3
delete/update ST		
min/max for 9 cols	8.934	9.121
min/max for 5 cols	5.082	5.170
min/max for 1 col	899	1.108

**Table 1. Performance RDBMS functionality in msec**

In Table 1 the performance of the RDBMS components, the stored procedures and triggers, are presented. *SP* labels stored procedures, *RT* row-wise triggers and *ST* statement triggers. The initial creation time for the basic data statistics is

nearly linear (2 times more rows need 2.49 times longer ) and occurs just once for each monitored table. The values for the row-wise triggers show that they are independent of the table size (in terms of numbers of rows). The most expensive part is to keep the min and max values for each column up-to-date. Each line in the bottom of Table 1 shows the values for the worst case (after the delete/update of a row which holds the current min/max value(s)) of one, five or nine column(s) which then have to be refined. Otherwise, the delete/update statement trigger returns immediately without any processing.



**Figure 9. Overall time of the developed DPP activities**

## 6 Related Work

A well known Grid project related to data pre-processing is the DataCutter framework [2], which carries out a rich set of queries and application specific data transformations on data streams. It is a more generic framework for developing distributed filters than our approach, which is focused on supporting data mining with tailored statistics and DPP methods for row-based data sets. [9] already explored RDBMS capabilities for data mining and used them to perform scalable classification. We are not using the RDBMS for applying DPP methods, instead we use it to provide continuous data statistics. [13] provides incrementally maintained general purpose database summarization in contrast to our specialized statistics to support the data mining process. [6] presents an algorithm to dynamically compute quantiles, which seems promising in order to provide continuous advanced data statistics inside the RDBMS.

## 7 Conclusions

The quality of integrated data sources can only be as *good* as their constituent parts. A lot of data mining techniques have *high requirements on the cleanness* of the input data. Having in mind how time consuming DPP is and what a great impact it has *not only* on the outcome of a knowledge discovery process [12], there is an urgent need for an integrated, flexible and easy to (re-)use solution. The extensions to the OGSA-DAI GDS presented in this paper allow to remotely work with query results and help to decide upon possible query candidates (with the help of the continuous data statistics). The continuous basic data statistics about a database already add value towards the metadata provided by a GDS and actively support the data mining process. The *standard based* (OGSA-DAI, PMML and Webrowset format) service implementation of D<sup>3</sup>G, allowing *flexible usage* of the DPP functionality (which is easily *extendable* with user defined DPP functions), together with the RDBMS exploiting part of D<sup>3</sup>G, with non-expensive exact continuous data statistics for specific tables, fulfill important requirements on a DPP service. Our future work agenda includes the following topics: further performance testing and tuning (especially the effect of the number of numerical columns on the performance), investigate and implement support for applying DPP methods in passive usage mode on the service side as well as extending the

RDBMS implementation towards the support of advanced data statistics.

## Acknowledgments

This work is being supported by the research projects "Modern Data Analysis on Computational Grids", "Relational machine learning for analysis of biomedical data" and "SemDIG" – funded by Austrian and Czech research foundations. We want to thank the attendees of the First DIALOGUE Workshop 2005 in Columbus, Ohio for valuable comments and discussions.

## References

- [1] PREDO. <http://cyber.felk.cvut.cz/gerstner/machine-learning/sw/predo/>.
- [2] M. Beynon, R. Ferreira, T. Kurc, A. Sussman, and J. Saltz. DataCutter: Middleware for Filtering Very Large Scientific Datasets on Archival Storage Systems, 2000.
- [3] P. Brezany, A. M. Tjoa, M. Rusnak, and I. Janciak. Knowledge grid support for treatment of traumatic brain injury victims. In *Proceedings of the ICCSA*, Montreal, Canada, 2003.
- [4] Data Mining Group. PMML 3.0 Statistics. <http://www.dmg.org/v3-0/Statistics.html>.
- [5] GGF Database Access and Integration Services Working Group. <https://forge.gridforum.org/projects/dais-wg>.
- [6] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *Proceedings of the VLDB*, China, 2002.
- [7] J. Gray. Distributed computing economics. MSR-TR-2003-24, 2003.
- [8] W. Kim, B.-J. Choi, E.-K. Hong, S.-K. Kim, and D. Lee. A taxonomy of dirty data. *Data Min. Knowl. Discov.*, 7(1):81–99, 2003.
- [9] H. Lu and H. Liu. Decision tables: Scalable classification exploring RDBMS capabilities. In *Proceedings of the VLDB*, Egypt, 2000.
- [10] OGSA-DAI. <http://www.ogsadai.org>.
- [11] D. Pyle. *Data preparation for data mining*. Morgan Kaufmann, 1999.
- [12] T. C. Redman. The impact of poor data quality on the typical enterprise. *Commun. ACM*, 41(2):79–82, 1998.
- [13] R. Saint-Paul, G. Raschia, and N. Mouaddib. General purpose database summarization. In *Proceedings of the VLDB*, Norway, 2005.
- [14] SUN. Webrowset schema. <http://java.sun.com/xml/ns/jdbc/webrowset.xsd>.
- [15] University of Vienna. GridMiner project. <http://www.gridminer.org>.
- [16] P. Watson. Databases in grid applications: Locality and distribution, 2005.